EXHIBIT 17

tari-project / tari Public

The Tari protocol

tari.com

BSD-3-Clause license

283 stars 534 forks

Star Notifications

Code Issues 84 Pull requests 14 Discussions Actions Projects 1 Security In

development Go to file

sdbondi tests: ignores peer seed test that requires external DNS (#5187) ... ✗ yesterday 6,140

View code

README.md

PASSED

The Tari protocol

A number of applications have been developed by the Tari community to implement the Tari protocol. These are:

- Tari Base Node
- Tari Console Wallet
- Tari Miner
- Tari Merge Mining Proxy
- Tari Aurora wallets for Android and iOS

Only the first four applications will be discussed in this README (see wallet-android and wallet-ios for mobile wallets' repos).

Installing using binaries

Download

Download binaries from tari.com. This is the easiest way to run a Tari node, but you're essentially trusting the person that built and uploaded them that nothing untoward has happened.

Hashes of the binaries are available alongside the downloads. You can get the hash of your download by opening a terminal or command prompt and running the following:

(*nix)

```
shasum -a256 <PATH_TO_BINARY_INSTALL_FILE>
```

(Windows)

```
certUtil -hashfile <PATH_TO_BINARY_INSTALL_FILE> SHA256
```

If the result doesn't match the published hash, don't run the binary. Note that this only checks that your binary was downloaded correctly; it cannot detect if the binary was replaced by a bad actor. If you need to ensure that your binary matches the source, see Building from source below.

Install

After you have downloaded the binaries, you need to install them. This is easy to do, and works as follows:

On *Nix

Assuming you want to install the Tari applications into your home folder, then, from within a terminal:

```
cd ~
tar -xf <PATH_TO_BINARY_INSTALL_FILE>
```

After this, the Tari applications will be located in `~/tari_esmeralda_testnet` with a selection of soft links to run them.

On Windows

Just double-click the installer and accept all the default prompts. The Tari applications will be located in the folder you selected during installation, and can be run by double-clicking the various shortcuts or via the Windows menu (`Tari Testnet`).

Runtime links

Use the one-click miner

Execute the `start_all` soft link/shortcut; this will start everything you need depending on the choices you make when prompted:

- Tor services started by default
- Tari Base Node, or
- Tari Base Node & Tari Console Wallet, or
- Tari Base Node & Tari Console Wallet & Tari Miner, or
- Tari Base Node & Tari Console Wallet & Tari Merge Mining Proxy & XMRig

Start all applications individually

- Execute the `start_tari_base_node` soft link/shortcut; this will also start the Tor services if not running already that needs to be running before the base node can run (do not close the Tor console).
- Execute the `start_tari_console_wallet` soft link/shortcut; this will also start the Tor services that needs to be running before the base node can run (do not close the Tor console).

  Note: The Tor console will output `[notice] Bootstrapped 100% (done): Done` when the Tor services have fully started.
- Depending on your choice of mining:
  - SHA3 stand-alone mining
    - Execute the `start_tari_miner` soft link/shortcut.
  - Merge mining with Monero
    - Execute the `start_tari_merge_mining_proxy` soft link/shortcut.
    - Execute the `start_xmrig` shortcut.

Building from source

To build the Tari codebase from source, there are a few dependencies you need to have installed.

Install development packages

First you'll need to make sure you have a full development environment set up:

(macOS)

```
brew update
brew install cmake openssl tor coreutils automake
brew install --cask powershell
```

(macOS M1 chipset)

It is important to note that RandomX does not work on Xcode version 14.1 and newer. To compile Tari and run properly you need to run XCode version 14.0 or earlier. To run multiple versions of XCode you can use this guide here

If randomX unit tests are still failing, please update the Mac to ensure its running at least `Darwin Kernel Version 22.3.0`

(Ubuntu 18.04, including WSL-2 on Windows)

```
sudo apt-get update
sudo apt-get -y install openssl libssl-dev pkg-config libsqlite3-dev clang git cmake libc++-dev
libc++abi-dev libprotobuf-dev protobuf-compiler libncurses5-dev libncursesw5-dev
sudo apt-get install -y wget apt-transport-https
sudo wget -q "https://packages.microsoft.com/config/ubuntu/$(lsb_release -rs)/packages-microsoft-
prod.deb"
sudo dpkg -i packages-microsoft-prod.deb
sudo apt-get update
sudo add-apt-repository universe
sudo apt-get install -y powershell
```

(Windows)

First you'll need to make sure you have a full development environment set up:

- LLVM
  - https://releases.llvm.org/
  - Create a `LIBCLANG_PATH` environment variable pointing to the LLVM lib path, e.g.

    ```
    setx LIBCLANG_PATH "C:\Program Files\LLVM\lib"
    ```

- Build Tools
  - CMake (Used for RandomX)
  - Either:
    - Microsoft Visual Studio Version 2019 or later
      - C++ CMake tools for Windows
      - MSVC build tools (latest version for your platform ARM, ARM64 or x64.x86)
      - Spectre-mitigated libs (latest version for your platform ARM, ARM64 or x64.x86)

    or

    - Build Tools for Visual Studio 2019
- Perl for OpenSSL:
  - OpenSSL is compiled and statically linked by the included rust-openssl crate
  - Perl is required to compile this source on Windows, please download and install StrawberryPerl
- Tor
  - Download Tor Windows Expert Bundle
  - Extract to local path, e.g. `C:\Program Files (x86)\Tor Services`
  - Ensure the directory containing the Tor executable, e.g. `C:\Program Files (x86)\Tor Services\Tor`, is in the path

Install Rust (*nix)

You can follow along at The Rust Website or just follow these steps to get Rust installed on your machine.

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Then make sure that `cargo` has been added to your path.

```
export PATH="$HOME/.cargo/bin:$PATH"
```

Install Rust (Windows 10)

Follow the installation process for Windows at The Rust Website. Then make sure that `cargo` and `rustc` has been added to your path:

```
cargo --version
rustc --version
```

Checkout the source code

In your directory of choice (e.g. `%USERPROFILE%\Code` on Windows), clone the Tari repo

```
git clone https://github.com/tari-project/tari.git
```

Build

Grab a cup of coffee and begin the Tari build

(*nix)

```
cd tari
cargo build --release
```

(Windows)

This is similar to building in Ubuntu, except the Microsoft Visual Studio environment must be sourced. Open the appropriate *x64\x86 Native Tools Command Prompt for VS 2019*, and in your main Tari directory perform the build, which will create the executable inside your `%USERPROFILE%\Code\tari\target\release` directory:

```
cd %USERPROFILE%\Code\tari
cargo build --release
```

A successful build should output something as follows

```
   Compiling tari_wallet v0.0.9 (.../tari/base_layer/wallet)
   Compiling test_faucet v0.0.1 (.../tari/applications/test_faucet)
   Compiling tari_wallet_ffi v0.0.9 (.../tari/base_layer/wallet_ffi)
   Compiling tari_base_node v0.0.9 (.../tari/applications/tari_base_node)
    Finished release [optimized] target(s) in 12m 24s
```

Compiled executable can be found by following path:

```
./target/release/tari_base_node
./target/release/tari_console_wallet
./target/release/tari_merge_mining_proxy
./target/release/tari_miner
```

Alternatively, cargo can build and install the executable into `~/.cargo/bin` (`%USERPROFILE%\.cargo\bin` on Windows), so it will be executable from anywhere on your system.

```
cargo install --path=applications/tari_base_node --force
cargo install --path=applications/tari_console_wallet --force
cargo install --path=applications/tari_merge_mining_proxy --force
cargo install --path=applications/tari_miner --force
```

Alternatively, cargo can build and install the executable into `%USERPROFILE%\.cargo\bin`, so it will be executable from anywhere on your system.

```
cargo install --path=applications/tari_base_node --force
cargo install --path=applications/tari_console_wallet --force
cargo install --path=applications/tari_merge_mining_proxy --force
cargo install --path=applications/tari_miner --force
```

Run

The executables will either be inside your `~/tari/target/release` (on Linux) or `%USERPROFILE%\Code\tari\target\release` (on Windows) directory, or alternatively, inside your `~/.cargo/bin` (on Linux) `%USERPROFILE%\.cargo\bin` (on Windows) directory, depending on the build choice above, and must be run from the command line. If the former build method was used, you can run it from that directory, or you more likely want to copy it somewhere more convenient. Make sure to start Tor service `~/tari/applications/tari_base_node/osx/start_tor` (on Mac), `~/tari/applications/tari_base_node/linux/start_tor` (on Linux) or `%USERPROFILE%\Code\tari\applications\tari_base_node\windows\start_tor.lnk` (on Windows).

To run from any directory of your choice, where the executable is visible in the path (first time use):

```
tari_base_node --init
tari_base_node

tari_console_wallet --init

tari_merge_mining_proxy

tari_miner --init
```

Consecutive runs:

```
tari_base_node

tari_console_wallet

tari_merge_mining_proxy

tari_miner
```

Alternatively, you can run the Tari applications from your source directory using `cargo`, and just omit the `--release` flag if you want to run in debug mode (first time use):

```
cargo run --bin tari_base_node --release --  --init
cargo run --bin tari_base_node --release

cargo run --bin tari_merge_mining_proxy --release

cargo run --bin tari_console_wallet --release --  --init

cargo run --bin tari_miner --release
```

Consecutive runs:

```
cargo run --bin tari_base_node --release

cargo run --bin tari_console_wallet --release

cargo run --bin tari_merge_mining_proxy --release

cargo run --bin tari_miner --release
```

Using all the default options, the blockchain database, wallet database, console wallet database, log files and all configuration files will be created in the `~/.tari` (on Linux) or `%USERPROFILE%\.tari` (on Windows) directory. Alternatively, by specifying `--base-path <base-path>` on the command line as well, all of this will be created in that directory.

Advanced build configurations

- Vagrant: See Building with Vagrant, using Vagrant to build and run a basenode, as cleanly as possible.

Using Docker

Running the base node with a docker image

Tari Base Node Docker images can be found at https://quay.io/repository/tarilabs/tari_base_node

Using `docker-compose.yaml`

```
version: "3"
```

```
services:
  tari_base_node:
    image: quay.io/tarilabs/tari_base_node:v0.5.4
    restart: unless-stopped
    volumes:
      - ./data:/root/.tari
# These 2 params are required for an interactive docker-compose session
    stdin_open: true
    tty: true
    expose:
      - 18142
    ports:
      - "18142:18142"
```

Then run `docker-compose up -d` to start your docker service.

Check the running state with `docker-compose ps`

```
        Name            Command    State            Ports
------------------------------------------------------------------
tbn_tari_base_node_1   start.sh   Up      0.0.0.0:18142->18142/tcp
```

To connect to the console, use `docker ps` to get the container ID which to attach to the tari_base_node in docker

```
CONTAINER ID        IMAGE                                    COMMAND             CREATED
STATUS              PORTS                          NAMES
73427509a4bb        quay.io/tarilabs/tari_base_node:v0.5.4   "start.sh"          45 minutes ago
Up 26 minutes       0.0.0.0:18142->18142/tcp   tbn_tari_base_node_1
```

With the container ID `73427509a4bb`, connect to the tari_base_node console as follows `docker attach 73427509a4bb`

```
>> help
Available commands are:
help, version, get-chain-metadata, list-peers, reset-offline-peers, ban-peer, unban-peer, list-
connections, list-headers,
check-db, calc-timing, discover-peer, get-block, search-utxo, search-kernel, search-stxo, get-
mempool-stats,
get-mempool-state, whoami, get-state-info, quit, exit
>> get-chain-metadata
Height of longest chain : 5228
Geometric mean of longest chain : 5892870
Best block : 2c4f92854b2160324b8afebaa476b39be4004d2a7a19c69dd2d4e4da257bfee2
Pruning horizon : 0
Effective pruned height : 0
>> get-state-info
Current state machine state:
Synchronizing blocks: Syncing from the following peers:
510c83279adc7cb7d7dda0aa07
Syncing 5229/5233
```

Building a docker image

If you don't want to use the docker images provided by the community, you can roll your own!

First, clone the Tari repo

```
git clone git@github.com:tari-project/tari.git
```

Then build the image using the dockerfile in `buildtools`. The base node docker file build the application and then places the binary inside a small container, keeping the executable binary to a minimum.

```
docker build -t tari_base_node:latest -f ./buildtools/base_node.Dockerfile .
```

Test your image

```
docker run --rm -ti tari_base_node tari_base_node --help
```

Run the base node

```
docker run -ti -v /path/to/config/dir:/root/.tari tari_base_node
```

Default docker builds for base x86-64 CPU. Better performing builds can be created by passing build options

```
docker build -t tari_base_node:performance --build-arg TBN_ARCH=skylake --build-arg
TBN_FEATURES=avx2 -f ./buildtools/base_node.Dockerfile .
```

Mining

The Tari protocol supports hybrid mining; stand-alone or pooled SHA3 mining using the Tari Miner or merged mining with Monero using the Tari Merge Mining Proxy in conjunction with XMRig (RandomX-based mining). Blocks to be won by stand-alone and pooled SHA3 mining has been apportioned to approximately 40% and with Monero merged mining to approximately 60%. This apportionment is deeply baked into the Tari protocol and part of the consensus rules. The 40/60 split is determined by slightly different block target times for each algorithm, that when combined will give an average block time of approximately 120s. Each mining algorithms make use of Linear Weighted Moving Average (LWMA) maths to gracefully adjust the target difficulties to adhere to the respective target block times. Any block won by either mining algorithm will be accepted, and when there is a tie a geometric mean calculation will be used to decide the winner. This system is completely fair without any additional empirical meddling to try force a certain outcome.

Tari SHA3 mining

In order to perform SHA3 mining with Tari, the following applications are needed:

- A Tari Base Node [*to supply blockchain metadata information*];
- A Tari Console Wallet [*to collect the Tari block rewards (coinbase transactions)*];
- A Tari Miner [*to perform the mining*];

In order to perform pooled SHA3 mining with Tari, the following applications are needed:

- For a pool operator:
  - A Tari Base Node [*to supply blockchain metadata information*];
  - A Tari Console Wallet [*to collect the Tari block rewards (coinbase transactions)*];
  - Miningcore [*pool software supporting various cryptocurrencies, configured for Tari*]
- For a miner:
  - A Tari Console Wallet [*to collect the share rewards (pool payouts)*];
  - A Tari Miner [*to perform the mining*];

Runtime prerequisites

The Tari Base Node, Tari Console Wallet and Tari Miner can all run in the same directory. By performing the default installation as described in Installing using binaries, all these applications will be available.

For MiningCore see the Linux and Windows build instructions.

Configuration prerequisites

The configuration prerequisites are the same for all four Tari applications. After performing a default installation, locate the main configuration file (`config.toml`), which will be created in the `~/tari_esmeralda_testnet/config` (on Linux) or `%USERPROFILE%\.tari-testnet\config` (on Windows) directory.

With the main configuration file, in addition to the settings already present, the following must also be enabled for the Tari Base Node and the Tari Console Wallet, if they are not enabled already. Under sections **`base_node.esmeralda`** and **`wallet`** respectively:

```
[wallet]

grpc_address = "127.0.0.1:18143"
```

```
[base_node.esmeralda]
transport = "tor"
allow_test_addresses = false
grpc_enabled = true
grpc_base_node_address = "127.0.0.1:18142"
```

For MiningCore:

See example configuration here.

For the Tari Miner there are some additional settings under section **`miner`** that can be changed:

- For SHA3 Mining:

```
[miner]
# Number of mining threads
# Default: number of logical CPU cores
#num_mining_threads=8

# GRPC address of base node
# Default: value from `base_node.grpc_base_node_address`
#base_node_grpc_address = "127.0.0.1:18142"

# GRPC address of console wallet
# Default: value from `wallet.grpc_address`
#wallet_grpc_address = "127.0.0.1:18143"

# Start mining only when base node is bootstrapped
# and current block height is on the tip of network
# Default: true
#mine_on_tip_only=true

# Will check tip with node every N seconds and restart mining
# if height already taken and option `mine_on_tip_only` is set
# to true
# Default: 30 seconds
#validate_tip_timeout_sec=30
```

For pooled SHA3 mining:

```
[miner]
# Number of mining threads
# Default: number of logical CPU cores
#num_mining_threads=8

# Stratum Mode configuration
# mining_pool_address = "miningcore.tari.com:3052"
# mining_wallet_address = "YOUR_WALLET_PUBLIC_KEY"
# mining_worker_name = "worker1"
```

Uncomment `mining_pool_address` and `mining_wallet_address`. Adjust the values to your intended configuration. `mining_worker_name` is an optional configuration field allowing you to name your worker.

Perform SHA3 mining

- For SHA3 mining: Tor and the required Tari applications must be started and preferably in this order:
  - Tor:
    - Linux/OSX: Execute `start_tor.sh`.
    - Windows: `Start Tor Serviecs` menu item or `start_tor` shortcut in the Tari installation folder.
    - Tari Base Node:
    - Linux/OSX: As per Runtime links.

        - Windows: As per Runtime links or `Start Base Node` menu item or `start_tari_base_node` shortcut in the Tari installation folder.

    - Tari Console Wallet:

        - Linux/OSX: As per Runtime links.
        - Windows: As per Runtime links or `Start Console Wallet` menu item or `start_tari_console_wallet` shortcut in the Tari installation folder.

    - Tari Miner:

        - Linux/OSX: As per Runtime links.
        - Windows: As per Runtime links or `Start Miner` menu item or `start_tari_miner` shortcut in the Tari installation folder.

Look out for the following types of messages on the Tari Miner console to confirm that it is connected properly and performing mining:

```
2021-02-26 11:24:23.604202000 [tari_miner] INFO  Connecting to base node at http://127.0.0.1:18151
2021-02-26 11:24:23.606260800 [tari_miner] INFO  Connecting to wallet at http://127.0.0.1:18161
2021-02-26 11:24:23.721890400 [tari_miner::miner] INFO  Mining thread 0 started
2021-02-26 11:24:23.722287800 [tari_miner::miner] INFO  Mining thread 1 started
2021-02-26 11:24:23.722505500 [tari_miner::miner] INFO  Mining thread 2 started
2021-02-26 11:28:19.687855700 [tari_miner::miner] INFO  Mining thread 2 stopped
2021-02-26 11:28:19.688251200 [tari_miner] INFO  Miner 2 found block header BlockHeader { hash:
[...], version: 1,
  height: 8493, prev_hash: [...], timestamp: Some(Timestamp { seconds: 1614331698, nanos: 0 }),
output_mr: [...],
  witness_mr: [...], total_kernel_offset: [...], nonce: 8415580256943728281, pow: Some(ProofOfWork
{ pow_algo: 2,
  pow_data: [] }), kernel_mmr_size: 24983, output_mmr_size: 125474 } with difficulty 7316856839
```

- For pooled SHA3 Mining:

    - Pool Operators: Tor and the required Tari applications must be started in this order:

        - Tor:

            - Linux/OSX: Execute `start_tor.sh`.
            - Windows: `Start Tor Serviecs` menu item or `start_tor` shortcut in the Tari installation folder.

        - Tari Base Node:

            - Linux/OSX: As per Runtime links.
            - Windows: As per Runtime links or `Start Base Node` menu item or `start_tari_base_node` shortcut in the Tari installation folder.

        - Tari Console Wallet:

        - Linux/OSX: As per Runtime links.
        - Windows: As per Runtime links or `Start Console Wallet` menu item or `start_tari_console_wallet` shortcut in the Tari installation folder.
    - MiningCore
  - Miners:
    - Tari Miner:
        - Linux/OSX: As per Runtime links.
        - Windows: As per Runtime links or `Start Miner` menu item or `start_tari_miner` shortcut in the Tari installation folder.

Tari merge mining

In order to perform merge mining with Tari, the following applications are needed:

- A Tari Base Node [*to supply blockchain metadata information*];
- A Tari Console Wallet [*to collect the Tari block rewards (coinbase transactions)*];
- A Tari Merge Mining Proxy [*to enable communication between all applications*];
- XMRig [*to perform the mining*];
- Monero wallet (specifically a stagenet wallet address during testnet; the one provided can be used, or a custom one can be set up) [*to collect Monero block rewards (coinbase transactions)*].

The Tari Merge Mining Proxy will be the communication gateway between all these applications and will coordinate all activities. It will also submit finalized Tari and Monero blocks to the respective networks when RandomX is solved at the respective difficulties.

Runtime prerequisites

The Tari Base Node, Tari Console Wallet and Tari Merge Mining Proxy can all run in the same directory, whereas XMRig will run in its own directory. By performing the default installation as described in Installing using binaries, all these applications will be available.

XMRig can also be build from sources. If that is your preference, follow these instructions: https://xmrig.com/docs/miner/.

Configuration prerequisites

Tari applications

The configuration prerequisites are the same for all three Tari applications. After performing a default installation, locate the main configuration file (`config.toml`), which will be created in the `~/tari_esmeralda_testnet/config` (on Linux) or `%USERPROFILE%\.tari-testnet\config` (on Windows) directory.

With the main configuration file, in addition to the settings already present, the following must also be enabled if they are not enabled already:

- For the Tari Base Node and the Tari Console Wallet, under sections **`base_node.esmeralda`** and **`wallet`** respectively

```
[wallet]
grpc_address = "127.0.0.1:18143"
```

```
[base_node.esmeralda]
transpo*_r_*t = "tor"
allow_test_addresses = false
base_node_grpc_address = "127.0.0.1:18142"
```

And then depending on if you are using solo mining or self-select mining you will use one of the following:

Solo mining

- For the Tari Merge Mining Proxy, under section `merge_mining_proxy`

```
[merge_mining_proxy]
monerod_url = [ # stagenet
  "http://stagenet.xmr-tw.org:38081",
  "http://stagenet.community.xmr.to:38081",
  "http://monero-stagenet.exan.tech:38081",
  "http://xmr-lux.boldsuck.org:38081",
  "http://singapore.node.xmr.pm:38081",
]

proxy_host_address = "127.0.0.1:18081"
proxy_submit_to_origin = true
monerod_use_auth = false
monerod_username = ""
monerod_password = ""
```

Self-Select mining

- For the Tari Merge Mining Proxy, under section `merge_mining_proxy`

```
[merge_mining_proxy]
monerod_url = [ # stagenet
  "http://stagenet.xmr-tw.org:38081",
  "http://stagenet.community.xmr.to:38081",
  "http://monero-stagenet.exan.tech:38081",
  "http://xmr-lux.boldsuck.org:38081",
  "http://singapore.node.xmr.pm:38081",
]

proxy_host_address = "127.0.0.1:18081"
proxy_submit_to_origin = false
monerod_use_auth = false
monerod_username = ""
monerod_password = ""
```

Note: The ports `18081`, `18142` and `18143` shown in the example above should not be in use by other processes. If they are, choose different ports. You will need to update the ports in the steps below as well.

The `monerod_url` set must contain valid addresses (`host:port`) for `monerod` that is running Monero mainnet (e.g. `["http://18.132.124.81:18081"]`) or stagenet (e.g. `["http://monero-stagenet.exan.tech:38081"]`), which can be a public node or local instance. To test if the `monerod_url` address is working properly, try to paste `host:port/get_height` in an internet browser, for example:

```
http://18.132.124.81:18081/get_height
```

A typical response would be:

```
{
  "hash": "ce32dd0a6e3220d57c368f2cd01e5980a9b4d70f02b27274d67142d5b26cb4d6",
  "height": 2277206,
  "status": "OK",
  "untrusted": false
}
```

***Note:** A guide to setting up a local Monero stagenet on Linux can be found here.*

XMRig configuration

The XMRig configuration must be prepared for either solo or pool merged mining with Monero. It is advisable to use a configuration file for XMRig as this offers more flexibility, otherwise, the configuration parameters can be passed in via the command line upon runtime.

Notes:

- Monero mainnet and stagenet wallet addresses can only be used with the corresponding network. The `monerod_url` configuration setting (see Tari applications) must also correspond to the chosen network.
- For the solo mining configuration, Monero doesn't currently support requesting templates to mine on with the address being a subaddress. It is possible to do with the self-select configuration since the template is requested by the miner with the wallet address of the pool.

Solo-mining

The XMRig configuration wizard can be used to create a solo mining configuration file in JSON format:

- Start -> `+ New configuration`
- Pools -> `+ Add daemon`
  - With `Add new daemon for Solo mining`, complete the required information, then `+ Add daemon`:
    - `Host`, `Port`: This must correspond to the `proxy_host_address` in the Tari configuration file.
    - `Secure connection (TLS)`: Uncheck.
    - `Coin`: Monero.
    - `Wallet address`: This must be your own stagenet or mainnet wallet address, or you can use these donation addresses:

        - Public stagenet address at https://coin.fyi/news/monero/stagenet-wallet-8jyt89#! `55LTR8KniP4LQGJSPtbYDacR7dz8RBFnsfAKMaMuwUNYX6aQbBcovzDPyrQF9KXF9tVU6Xk3K8no1BywnJX6GvZX8yJsXvt`
        - Mainnet address `<Enter your own mainnet wallet address here>`

- Backends -> Select `CPU` (`OpenCL` or `CUDA` also possible depending on your computer hardware).
- Misc -> With `Donate` , type in your preference.
- Result -> With `Config file` , copy or download, than save as `config.json` .

Using the public stagenet wallet address above the resulting configuration file should look like this:

```
{
    "autosave": true,
    "cpu": true,
    "opencl": false,
    "cuda": false,
    "pools": [
        {
            "coin": "monero",
            "url": "127.0.0.1:18081",
            "user":
"55LTR8KniP4LQGJSPtbYDacR7dz8RBFnsfAKMaMuwUNYX6aQbBcovzDPyrQF9KXF9tVU6Xk3K8no1BywnJX6GvZX8yJsXvt",
            "tls": false,
            "daemon": true
        }
    ]
}
```

Pool mining with Self-Select

For pool mining, the configuration file obtained from the XMRig configuration wizard must be augmented with Tari specific settings. Using the wizard, create the following:

- Start -> `+ New configuration`
- Pools -> `+ Add pool` -> `Custom pool`
    - With `Add new custom pool` , complete the required information, then `+ Add pool` :
        - `Host` , `Port` : This must be for a Monero mainnet mining pool that supports the `self-select` .
        - `Secure connection (TLS)` : Check/Uncheck (based on the pool requirements).
        - `keepalive` : Check.
        - `nicehash` : Uncheck.
        - `User` : This must be your own mainnet wallet address, or you can use this address to donate to Monero:
            - Public mainnet address at https://www.getmonero.org/get-started/contributing/ `888tNkZrPN6JsEgekjMnABU4TBzc2Dt29EPAvkRxbANsAnjyPbb3iQ1YBRk1UXcdRsiKc9dhwMVgN5S9cQUiyoogDavup3H`
        - `Password` : A custom field that could be your wallet name or some other pool settings.

- `Coin`: Monero.
- `Algorithm`: rx/0.

- Backends -> Select `CPU` (`OpenCL` or `CUDA` also possible depending on your computer hardware).
- Misc -> With `Donate`, type in your preference.
- Result -> With `Config file`, copy or download, than save as `config.json`.
- Add custom entries for `"self-select": "127.0.0.1:18081"` and `"submit-to-origin": true` in the `"pools"` section.

Mining pool `cryptonote.social` requires you to add a personalized handle to the wallet address so that you can query your own pool statistics, separated by a full stop, i.e. `<YOUR WALLET ADDRESS>.<pool specific user name>`. For demonstration purposes, `donatemonero` has been associated with the public mainnet wallet address above. If you go to https://cryptonote.social/xmr and enter `donatemonero` in the `Username:` text box you will see some merge mining activity for that address. The configuration file used for this exercise is shown below:

```
{
    "autosave": true,
    "cpu": true,
    "opencl": false,
    "cuda": false,
    "pools": [
        {
            "coin": "monero",
            "algo": "rx/0",
            "url": "cryptonote.social:5555",
            "user":
"888tNkZrPN6JsEgekjMnABU4TBzc2Dt29EPAvkRxbANsAnjyPbb3iQ1YBRk1UXcdRsiKc9dhwMVgN5S9cQUiyoogDavup3H.dona
            "pass": "start_diff=220000;payment_scheme=pprop;donate=0.5",
            "tls": false,
            "keepalive": true,
            "nicehash": false,
            "self-select": "127.0.0.1:18081",
            "submit-to-origin": true
        }
    ]
}
```

Perform merge mining

Tor and the required Tari applications must be started and preferably in this order:

- Tor:
  - Linux/OSX: Execute `start_tor.sh`.
  - Windows: `Start Tor Serviecs` menu item or `start_tor` shortcut in the Tari installation folder.
- Tari Base Node:

- Linux/OSX: As per Runtime links.
- Windows: As per Runtime links or `Start Base Node` menu item or `start_tari_base_node` shortcut in the Tari installation folder.

- Tari Console Wallet:
  - Linux/OSX: As per Runtime links.
  - Windows: As per Runtime links or `Start Console Wallet` menu item or `start_tari_console_wallet` shortcut in the Tari installation folder.

- Tari Merge Mining Proxy:
  - Linux/OSX: As per Runtime links.
  - Windows: As per Runtime links or `Start Merge Mining Proxy` menu item or `start_tari_merge_mining_proxy` shortcut in the Tari installation folder.

In addition, select one of the merge mining options as outlined in solo or pool mining in the next paragraphs.

Solo merged mining with Monero

This paragraph is applicable to solo mining Monero on mainnet or stagenet and solo mining Tari on testnet.

Solo merged mining with Monero is supported using the `daemon` option.

Merge Mining Proxy configuration

As mentioned previously, the `monerod_url` field in the `config.toml` should be enabled for the corresponding mainnet or stagenet network Monero wallet address:

```
# URL to monerod
  monerod_url = [ # mainnet
  "http://18.132.124.81:18081",
  "http://xmr.support:18081",
  "http://node1.xmr-tw.org:18081",
  "http://xmr.nthrow.nyc:18081",
  ]
  monerod_url = [ # stagenet
    "http://stagenet.xmr-tw.org:38081",
    "http://stagenet.community.xmr.to:38081",
    "http://monero-stagenet.exan.tech:38081",
    "http://xmr-lux.boldsuck.org:38081",
    "http://singapore.node.xmr.pm:38081",
  ]
```

Runtime

Ensure the `config.json` configuration file discussed in Solo mining is copied to the XMRig build or install folder, then start XMRig:

- Linux/OSX: Execute `./xmrig` in the XMRig build or install folder.

- Windows: Execute `xmrig` in the XMRig build or install folder, or `Start XMRig` menu item or `start_xmrig` shortcut in the Tari installation folder.

  Note: On modern Windows versions, coin mining software is blocked by default, for example by Windows Defender. Ensure that these processes are allowed to run when challenged:

  - `PUA:Win32/CoinMiner`
  - `PUA:Win64/CoinMiner`
  - `App:XMRigMiner`

Look out for the following outputs in the XMRig console to confirm that it is connected to the Merge Mining Proxy and accepting jobs:

```
* POOL #1      127.0.0.1:18081 coin monero
```

```
[2021-01-21 12:10:18.960]  net      use daemon 127.0.0.1:18081  127.0.0.1
[2021-01-21 12:10:18.960]  net      new job from 127.0.0.1:18081 diff 286811 algo rx/0 height
756669
[2021-01-21 12:10:56.730]  cpu      rejected (0/1) diff 286811 "Block not accepted" (656 ms)
[2021-01-21 12:10:57.398]  net      new job from 127.0.0.1:18081 diff 293330 algo rx/0 height
756670
[2021-01-21 12:12:23.695]  miner    speed 10s/60s/15m 4089.0 4140.2 n/a H/s max 4390.9 H/s
[2021-01-21 12:12:57.983]  cpu      accepted (1/1) diff 293330 (594 ms)
```

The `cpu: rejected` and `cpu: accepted` messages originates from stagenet or mainnet `monerod`, and shows the Monero statistics. At this point, the mined and rejected Tari coinbases should be visible in the Tari Console Wallet.

Pool merged mining with Monero (self select)

This paragraph is applicable to pool mining Monero on mainnet and solo mining Tari on testnet.

Pool merged mining with Monero is supported using the Stratum mode self-select option via XMRig. Two mining pools we have tried out that support this feature are monero-pool, with its reference pool implementation running here, and cryptonote.social. With normal self select mode, XMRig requests a Monero block template from a third party and submits the solution to the mining pool. Tari added a `submit-to-origin` option to the self select mode whereby, if a solution has been found that only matches the pool difficulty, XMRig will submit the solution to the pool only, but if the achieved difficulty meets both that of the pool and Tari, it will be submitted to the Merge Mining Proxy as well as to the mining pool.

Merge Mining Proxy configuration

The `monerod_url` field in the `config.toml` should be enabled for the mainnet value:

```
# URL to monerod
  monerod_url = [ # mainnet
  "http://18.132.124.81:18081",
  "http://xmr.support:18081",
  "http://node1.xmr-tw.org:18081",
  "http://xmr.nthrow.nyc:18081",
  ]
```

Runtime

Ensure the `config.json` configuration file discussed in Pool mining with self select is copied to the XMRig build or install folder, then start XMRig as before for solo mining.

Look out for the following outputs in the XMRig console to confirm that it is connected to the pool and the Merge Mining Proxy and accepting jobs:

```
* POOL #1      cryptonote.social:5555 coin monero self-select 127.0.0.1:18081 submit-to-origin
```

```
[2021-01-18 11:40:48.392]  net      new job from cryptonote.social:5555 diff 220006 algo rx/0
height 2277084
[2021-01-18 11:41:22.378]  origin   submitted to origin daemon (1/0)  diff 284557 vs. 371742
[2021-01-18 11:41:22.812]  cpu      accepted (1/0) diff 220006 (433 ms)
[2021-01-18 11:41:39.201]  miner    speed 10s/60s/15m 1562.2 1630.4 n/a H/s max 1710.0 H/s
[2021-01-18 11:42:06.320]  cpu      accepted (2/0) diff 220006 (482 ms)
```

Status essages `origin: submitted to origin daemon (1/0)` and `origin: not submitted to origin daemon, difficulty too low (1/1)` pertains to submissions to the Tari network, and `cpu: accepted (1/0)` to the pool.

Mined and rejected Tari coinbases should be visible in the Tari Console Wallet, and pool shares in the pool interface. If you are using `cryptonote.social:5555` as in the example above, go to https://cryptonote.social/xmr and type in your wallet identity under `Username:` to see your shares, or try `taritest` if you used this configuration example.

Project documentation

- RFC documents are hosted on Github Pages. The source markdown is in the `RFC` directory.
- Source code documentation is hosted on docs.rs
- RFC repo

RFC documents

The RFCs are long-form technical documents proposing changes and features to the Tari network and ecosystem. They are hosted at https://rfc.tari.com, and the RFC repo is at https://github.com/tari-project/rfcs

Source code documentation

Run

```
cargo doc
```

to generate the documentation. The generated html sits in `target/doc/`. Alternatively, to open a specific package's documentation directly in your browser, run

```
cargo doc -p <package> --open
```

Code organisation

Out of date as of July 2022. TODO - Good first issue?

Conversation channels

We're generally on Discord

Releases

223 tags

Packages 7

- tor
- xmrig
- monerod

+ 4 packages

Used by 7

Contributors 41

+ 30 contributors

Languages

● Rust 96.3% ● C 1.3% ● Gherkin 1.0% ● Shell 0.8% ● JavaScript 0.2% ● Batchfile 0.2% ● Other 0.2%